VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

APRIL 2019

SMART HEALTHCARE SYSTEM

Project Report

under the guidance of

DR. MANJULA R.

submitted by

17BCE0440 SAMARTH ARORA

17BCE0472 APURV VAIBHAV

17BCE0932 ARJUN DIXIT

ABSTRACT:

The healthcare industry has been adopting innovative technologies that allow digitalization of health records and automation of clinical processes. The need for interoperability across different departments in healthcare requires a seamless data exchange inside the system. However, confidentiality and integrity of data are critical issues during the process of sharing data across different authorized parties. Hundreds of millions of medical records have been compromised in 2016 and the number increases. The emerging blockchain technology is a revolutionary mechanism, which ensures data integrity and confidentiality inside any system. Some healthcare providers have been inclined to implement blockchain technology as it presents a decentralized and encrypted way of storing and sharing information.

This emerging technology has a great potential to enhance the confidentiality and integrity of Electronic Health Records. In this paper, we conducted a systematic literature review to find out the research gaps and future research directions in blockchain technology in healthcare research. The literature is analyzed to find advantages, disadvantages and challenges of adopting blockchain technology in healthcare from people, process technology perspectives.

A blockchain is, in the simplest of terms, a time-stamped series of immutable record of data that is managed by a cluster of computers not owned by any single entity. Each of these blocks of data (i.e. block) are secured and bound to each other using cryptographic principles.

The reason why the blockchain has gained so much admiration is that:

- It is not owned by a single entity, hence it is decentralized
- The data is cryptographically stored inside
- The blockchain is immutable, so no one can tamper with the data that is inside the blockchain
- The blockchain is transparent so one can track the data if they want to

OBJECTIVE:

- This App enables users to express their symptoms and issues; with this they can consult a doctor online or offline. This application is fed with various symptoms and diseases associated.
- Also, patients can check their past medical records. In my project there are 4 facilities which are provided by me in the form of an android application.

- Marketplace. This is a place from where customer can buy the medicine by just entering the name of the medicine. He will get the shop where that medicine is available and he can order his medicine.
- Along with this he also come to know the distance of the shops which is far and which is near to him.
- On the other hand, shopkeeper can sign up to the app and after which he has to login and enter the medicine he has in his shop along with barcode and quantity.
- Authentication of Medicine, in this customer can enter the barcode of the medicine and he gets all the details from the medicine are manufactured and who is the supplier.
- For this there is also the sign up option for the factories, distributor and the supplier also who will enter the barcode of the medicine and the arrival time of the slot and departed time of the slot of that medicine.
- Each time they enter the details the app will form a block in the backend which will implement the block-chain.
- After this there is also an option for Health care in which the customer can search for the doctor and also get the free timing, location and the registration fee of the doctor. Doctor can also login to the app or if doctor would not use app we will fetch the details with the help of Google API's.
- There is also the facility of chat box and video calling. Patient can go for search with the help of voice search also. Customers and patient can also pay through online payment also if they want it do.

REQUIREMENT SPECIFICATIONS:

BACK – END REQUIREMENTS	DESCRIPTION
MongoDB	**The Language is to implement a data store that provides high performance, high availability, and automatic scaling. It is used as the database in our project to store the data .**
Express	**It is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile application and in our project we have used Express.js to provide a thin layer of fundamental web application features, without obscuring Node.js features that you know and love.**
NPM	**It is the package manager for the Node Java Script platform.**

	It puts modules in places so that Node can find them and manages dependency conflicts intelligently. It is extremely configurable to support a wide variety of use cases
jQuery	**jQuery is a JavaScript library designed to simplify HTML DOM tree traversal and manipulation, as well as event handling, CSS animation, and Ajax. It is free, open-source software using the permissive MIT License. We have used it as a fast and concise JavaScript Library that simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development**
Block chain	**A blockchain is a data structure that makes it possible to create a digital ledger of transactions and share it among a distributed network of computers. It uses cryptography to allow each participant on the network to manipulate the ledger in a secure way without the need for a central authority.**

FRONT - END REQUIREMENTS	**DESCRIPTION**
HTML	**Hypertext Markup Language is the standard markup language for creating web pages and web applications. With Cascading Style Sheets and JavaScript, it forms a triad of cornerstone technologies for the World Wide Web. Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.**
CSS	**Cascading Style Sheets is a style sheet language used for describing the presentation of a document written in a markup language like HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript. CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages**

Bootstrap	**Bootstrap is a free and open-source front-end web framework. It contains HTML and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions. Unlike many earlier web frameworks, it concerns itself with front-end development only.**
JavaScript	**JavaScript is a lightweight, interpreted programming language. It is designed for creating network-centric applications. It is complimentary to and integrated with Java. JavaScript is very easy to implement because it is integrated with HTML. It is open and cross-platform. The functioning of the front-end is controlled by JavaScript..**
Materialize CSS	**Materialize CSS is a UI component library which is created with CSS, JavaScript and HTML**

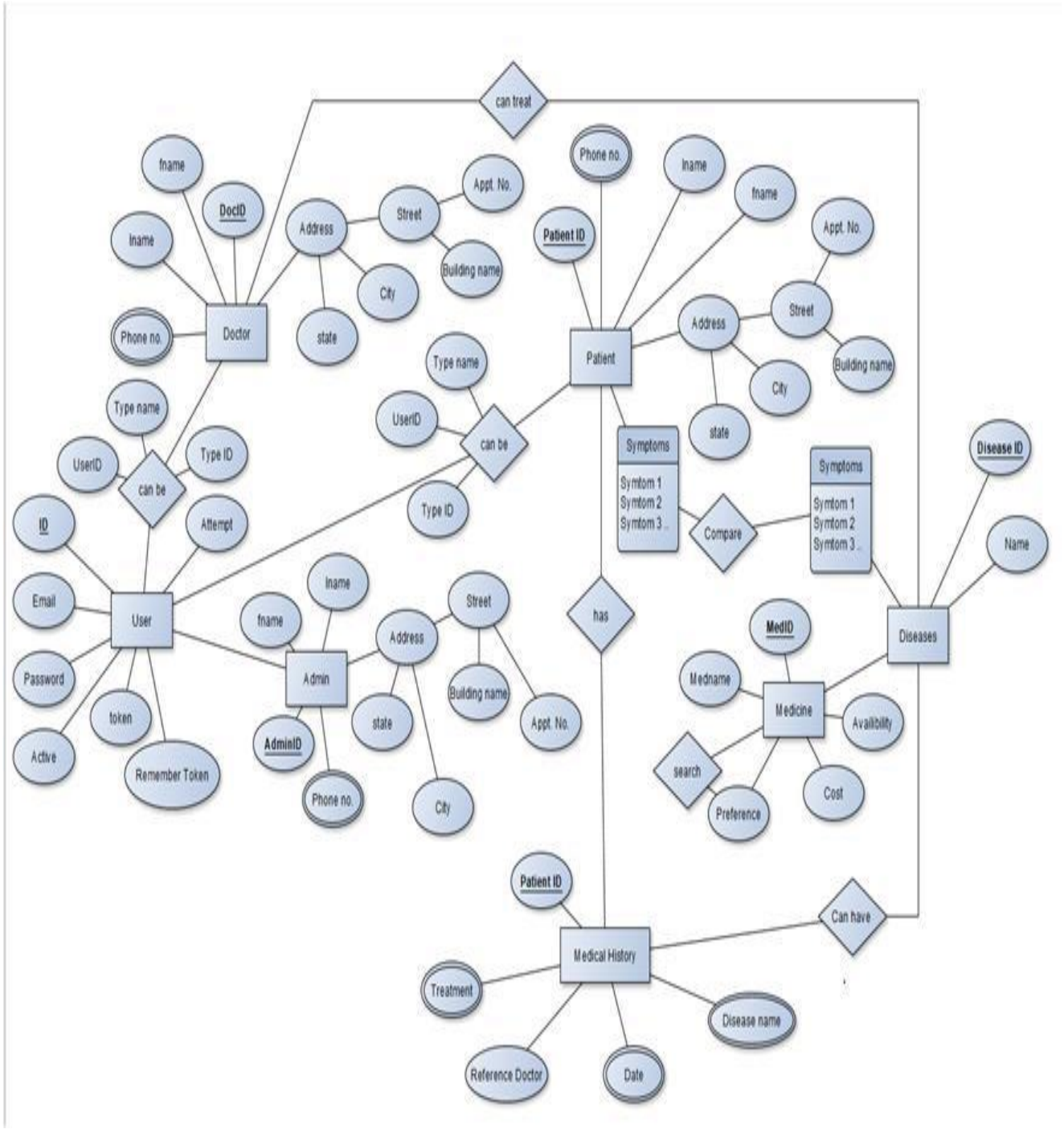
TECHNOLOGIES USED:

- Html
- CSS
- Bootstrap
- JavaScript
- jQuery
- Block chain
- Materialize CSS
- MongoDB
- Express
- Npm

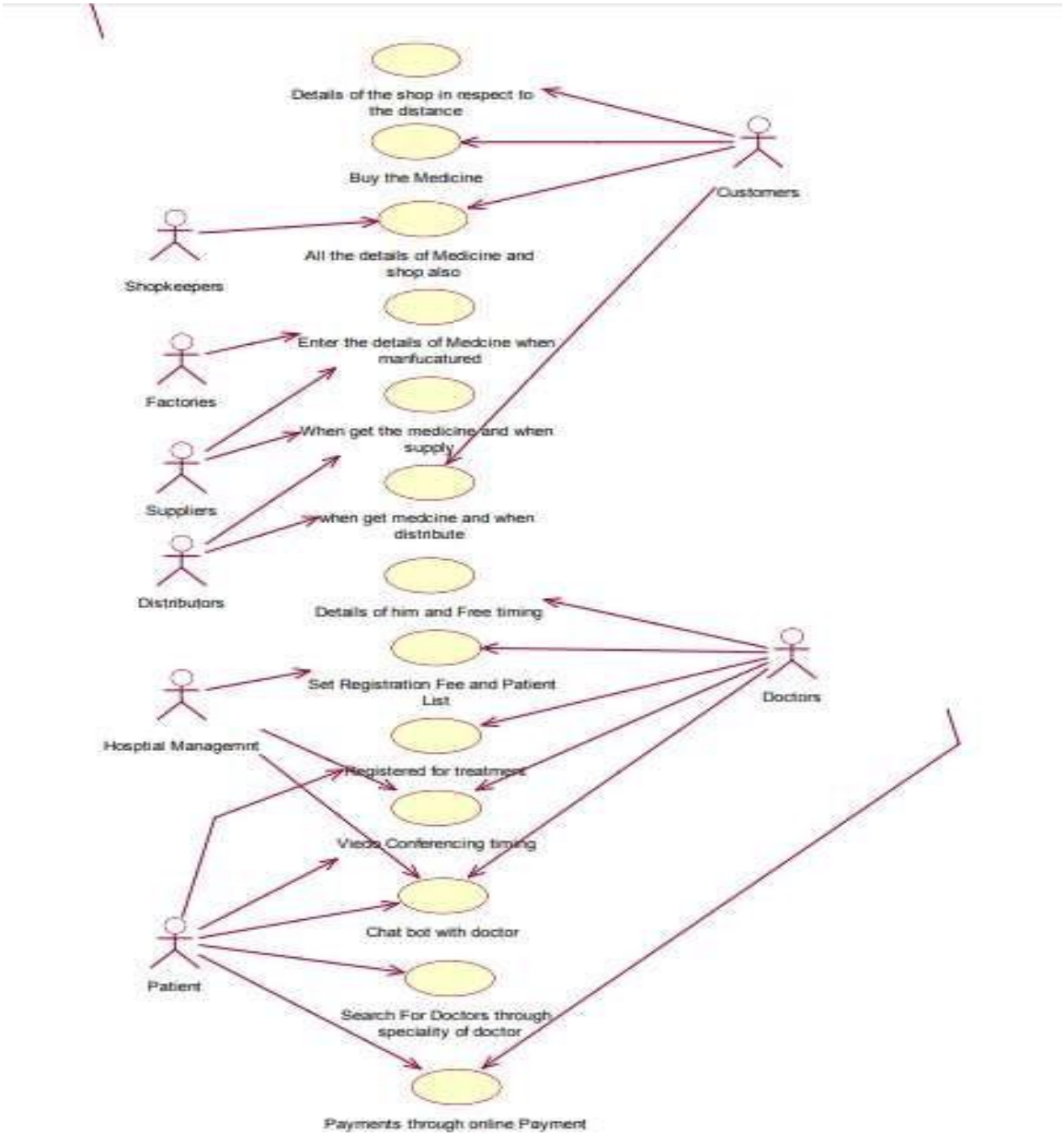
Functional Requirements:

1. Creating a Block in the Chain
2. Feeding the Data to Factory, Supplier and Distributer Units
3. Displaying the whole BlockChain
4. Searching for a Drug and Checking its Aunthenticity

Process Model



Use Case Diagram:



Software Process Model:

I have my projects requirements clear like I have 4 options to go with the app which are Producers, Health Center, Marketplace and Authentication of Medicine. Moreover all these options are independent of each other and I can prioritize the requirements (increments) to be delivered which is as follows:

1. Producers & Authentication of Medicine
2. Health Center & Marketplace

The requirement in each increment is also very clear and all as follows.

1. **Producers** increment have requirements which are sign up (for the first time), Login, Name of the factory/business man/shop, Location of respective authority,
Uniquely Identification Number, License Number, Medicine name, Place of Manufacture, Barcode, Quantity. Authentication of Medicine option is to check the Authenticity of a Medicine with the help of Barcode.

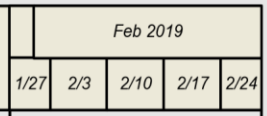
2. **Health Center** increment has requirements for searching the doctor (through symptoms or by just entering the name) and also registering the appointment at the specific time provided by the doctor or hospital. Marketplace increment has requirements which are Customer can buy medicine and shopkeeper can registered for their shops and home delivery facilities also.

The list of tasks to be performed is huge and clear. And early increment like
Producers (option 1) will act as prototype for Authentication of Medicine (option 2). This also has Lower risk of failure for overall project because I can deliver each increment one by one to the customer. And in each increment I go with waterfall model because every requirement is clear in each increment (which is mentioned above). Also customer would not want to wait for completing the project. So I will prefer

Process model based Gantt Chart:-

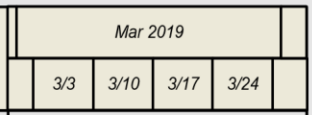
1. Producers & Authentication of Medicine

ID	Task Name	Start	Finish	Duration
1	Communication	1/29/2019	1/31/2019	.6w
2	Planning	1/31/2019	2/5/2019	.8w
3	Moduling	2/5/2019	2/18/2019	2w
4	Construction	2/18/2019	2/22/2019	1w
5	Deployment	2/22/2019	3/5/2019	1.5w



2. Health Center & Marketplace

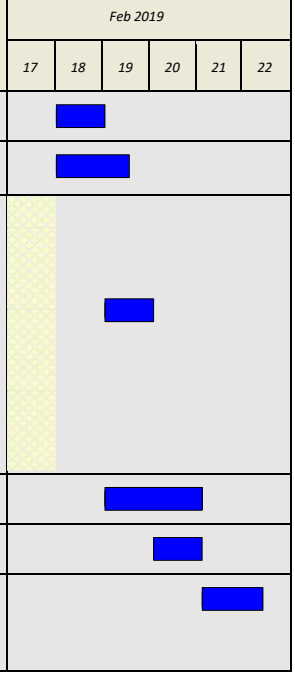
ID	Task Name	Start	Finish	Duration
1	Communication	2/28/2019	3/4/2019	.6w
2	Planning	3/4/2019	3/7/2019	.8w
3	Moduling	3/7/2019	3/20/2019	2w
4	Construction	3/20/2019	3/26/2019	1w
5	Deployment	3/26/2019	4/4/2019	1.5w



Product based Gantt chart: -

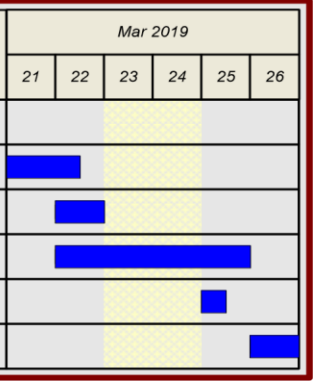
1. Producers & Authentication of Medicine

ID	Task Name	Start	Finish	Duration
1	Signup and Login	2/18/2019	2/18/2019	8h
2	Front-End for factory, supplier and Distributor	2/18/2019	2/19/2019	12h
3	Construction of Database for factory, supplier and Distributor	2/19/2019	2/19/2019	8h
4	Connection to the database	2/19/2019	2/20/2019	16h
5	Authentication Page of Medicine	2/20/2019	2/20/2019	8h
6	Connection of Authentication Page to database	2/21/2019	2/22/2019	10h

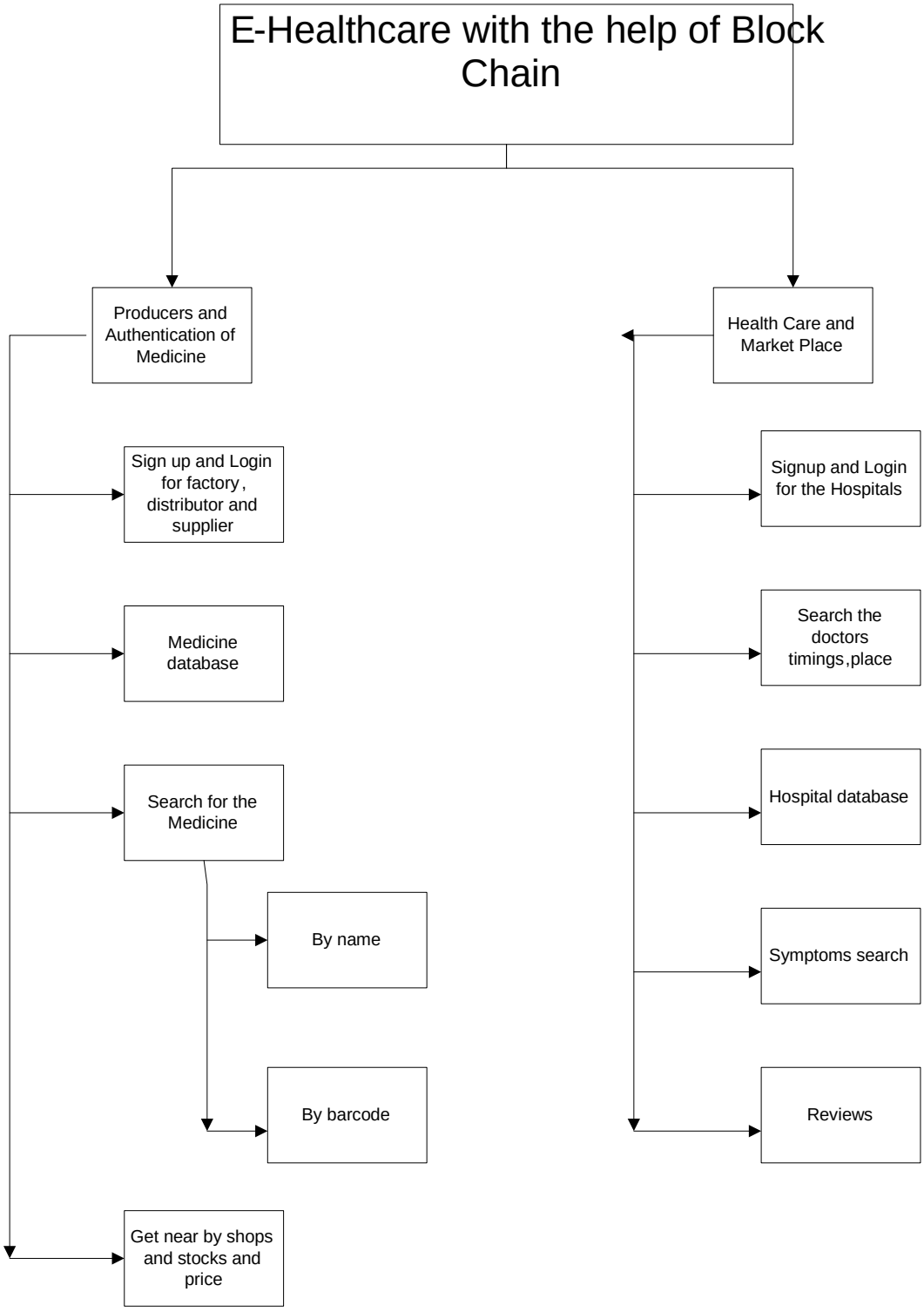


2. Health Center & Marketplace

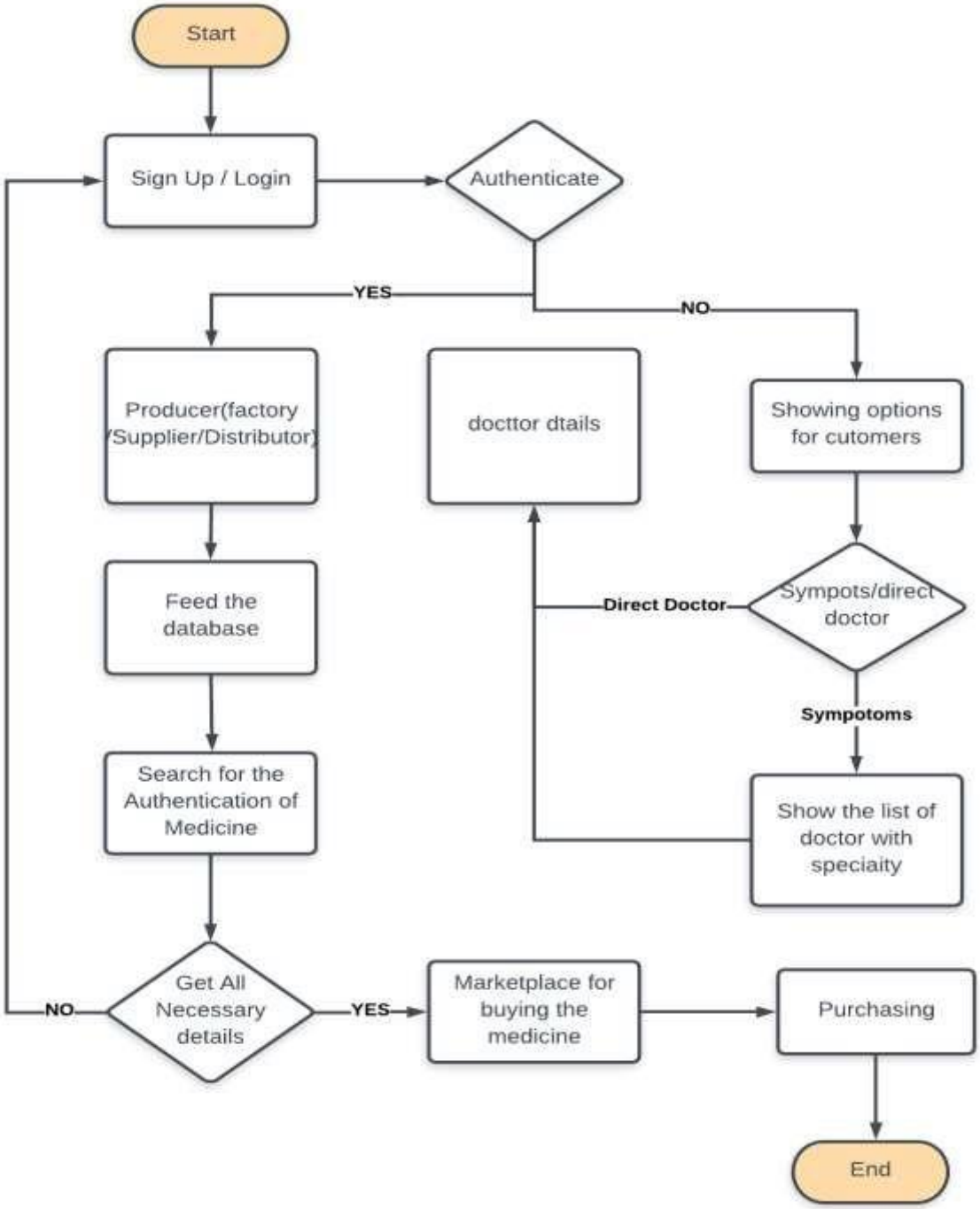
ID	Task Name	Start	Finish	Duration
1	Signup and Login for Hospitals	3/20/2019	3/20/2019	8h
2	Front-End for Hospitals and shopping	3/21/2019	3/22/2019	12h
3	Construction of Database	3/22/2019	3/22/2019	8h
4	Connection to the database	3/22/2019	3/25/2019	16h
5	Symptoms and Search for Medicine Page	3/25/2019	3/25/2019	4h
6	Backend of Search Page of Medicine	3/26/2019	3/26/2019	8h



Work breakdown Structure:



Activity Network:



Nonfunctional Requirements

Performance Requirements

The following are the performance requirements for the FCA Reservation Application:

Requirement	Description
System Response	The application must take no more than 4-7 seconds to respond to any request and return data from the ETA database.
System Responsiveness	The system must display a loading indicator to signify any requests or retrievals of data between the application, the ETA servers, and the ETA databases.
System Load Balancing	The system must be able to support 100 users without degradation. The system must support 500 users without any issue. However to upgrade the server capacity we can switch to the amazon aws,microsoft azure.
System User Session Timeout	The system will recognize when a user has been idle for a period of 4 minutes. At this time, the application will close and the user will be required to reopen the application to restore a secured session and log into the system.

Safety Requirements

The following are the safety requirements for the ETA:

Requirement	Description

System Accessibility	The application will react to the android phone accessibility settings for font size and font style. This includes default settings and special accommodations provided by Samsung for those with visual challenges.
Soft Color Scheme	The application will use a color scheme that does not include high contrast colors in order to reduce eye strain.
Smartphone Controls	The application will not override the safety functions of the android smartphone.
Lost or Low Battery	The application will close when the smartphone battery is depleted. This will require that the user re-open the application and restart a session after restoring the device's battery.
Lost or Stolen Device	the application and user accounts will have the capacity to be deactivated in cases where a user's device is lost or stolen.

Software Quality Attributes

The following are software quality attributes for the ETA:

Requirement	Description
Font Family	The application will use the android phone default font on all pages.
Font Size	Regular text will use the android phone default of 18-point font. The font size will increase to 24-point font for all headers and decrease to 12- point font for support information and footers.
Usability	The application will score no less than 7 on ease of use during usability tests.

Updates	Each update to the application will include versioning for troubleshooting and traceability purposes. The version number can be found on the About Page of the application.
Reusability	The application will be developed using Object Oriented programming. This platform allows for design features to be reused

CODE SNIPPETS

#Creation of a Block

```
const mongoose = require("mongoose");
const crypto = require("crypto");

const blockSchema = mongoose.Schema(
 {
  index: Number,
  data: Object,
  previousHash: String,
  hash: String,
  Content: String
 },
 {
  timestamps: true
 }
);

blockSchema.methods.calculateHash = function() {
 const hashString = JSON.stringify(this.data) + this.previousHash + this.index;
 const hash = crypto
  .createHash("md5")
  .update(hashString)
  .digest("hex");
 return hash;
};
```

```
module.exports = mongoose.model("Block", blockSchema);
```

#Creation of the whole BlockChain

```
const mongoose = require("mongoose");
const Promise = require("bluebird");
const Block = require("./block");

const blockChainSchema = mongoose.Schema({
  chain: [{
    type: mongoose.Schema.ObjectId,
    ref: "Block"
  }]
});

/**
 * @function addBlock
 * @param {Object} data
 */
blockChainSchema.methods.addBlock = function (data, variable) {
  console.log(variable);
  return this.model("Blockchain")
    .findOne()
    .populate("chain", "hash")
    .exec()
    .then(chain => {
      const index = this.chain.length;
      const previousHash = index !== 0 ? chain.chain[index - 1].hash : "0";
      let block = new Block({
        index: index,
        Content: variable,
        previousHash: previousHash,
        data: data
        // Content: Content.Content
      });
      block.hash = block.calculateHash();
      return block.save().then(savedBlock => {
        this.chain.push(savedBlock._id);
        return Promise.resolve(this);
```

```
    });
  })
  .catch(err => Promise.reject(err));
};

/**
 * @function validateChain
 */

blockChainSchema.methods.validateChain = function () {
 return this.model("Blockchain")
  .findOne()
  .populate("chain")
  .exec()
  .then(chain => {
   return Promise.map(chain.chain, (block, index) => {
    let outcome = "Valid";
    if (
     index > 0 &&
     chain.chain[index].previousHash !== chain.chain[index - 1].hash
    )
     outcome = "Invalid cause previous hash mismatch";
    return outcome;
   }).then(validities => {
    return Promise.resolve(validities);
   });
  })
  .catch(err => Promise.reject(err));
};

module.exports = mongoose.model("Blockchain", blockChainSchema);
```

#Running the WebApp on the local server

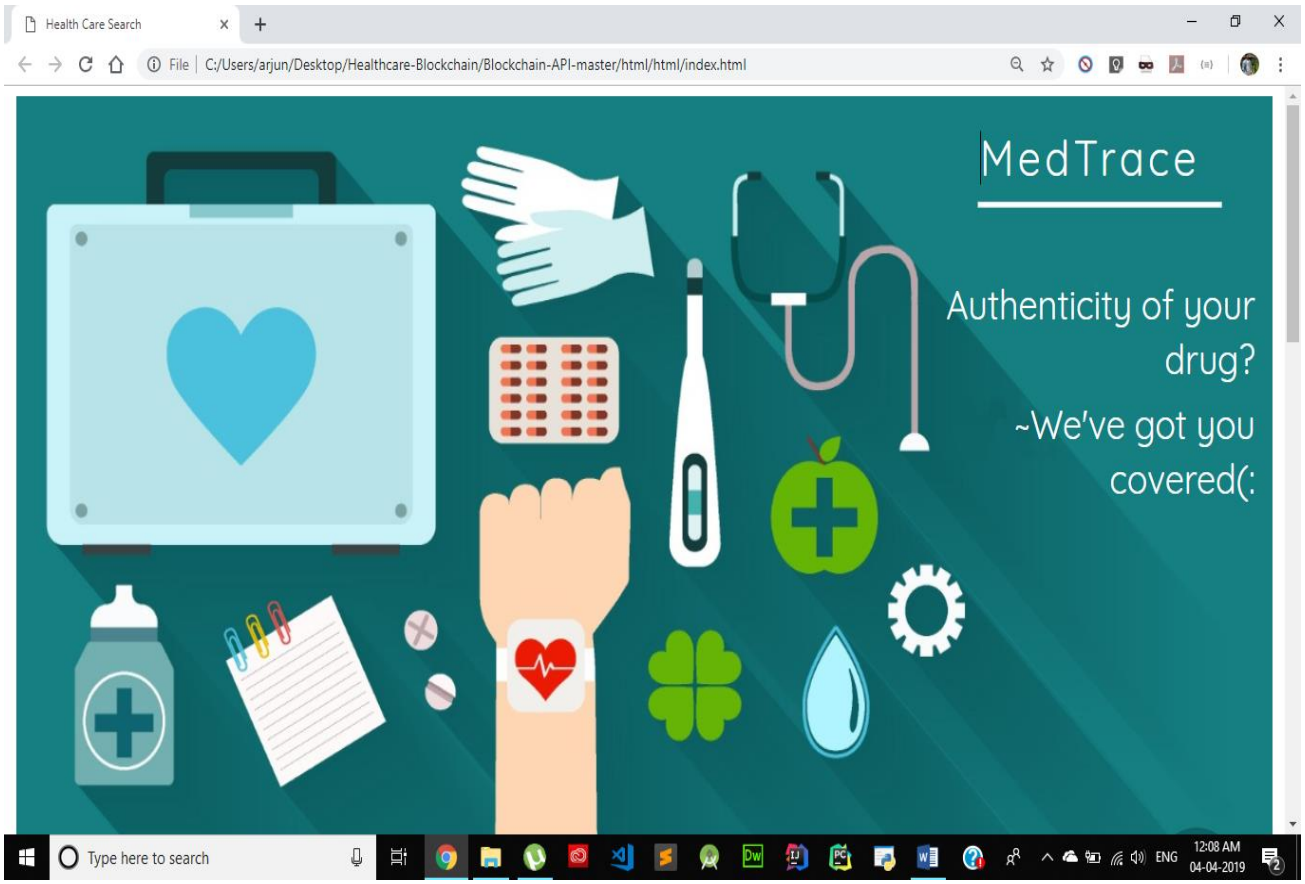
```
var createError = require("http-errors");
var express = require("express");
var path = require("path");
var cookieParser = require("cookie-parser");
```

```
var logger = require("morgan");
require("dotenv").config();
var indexRouter = require("./routes/index");

const mongoose = require("mongoose");
const Promise = require("bluebird");
mongoose.Promise = Promise;
mongoose.connect(
  "mongodb://localhost:27017/blockchain"
);

var app = express();

app.use(logger("dev"));
app.use(express.json());
app.use(
  express.urlencoded({
    extended: false
  })
);
app.use(cookieParser());
app.use(express.static(path.join(__dirname, "public")));

app.use("/", indexRouter);

// catch 404 and forward to error handler
app.use(function (req, res, next) {
  next(createError(404));
});

// error handler
app.use(function (err, req, res, next) {
  // set locals, only providing error in development
  res.locals.message = err.message;
  res.locals.error = req.app.get("env") === "development" ? err : {};

  return res.json({
    success: false,
    message: err.message,
    stack: err.stack
```

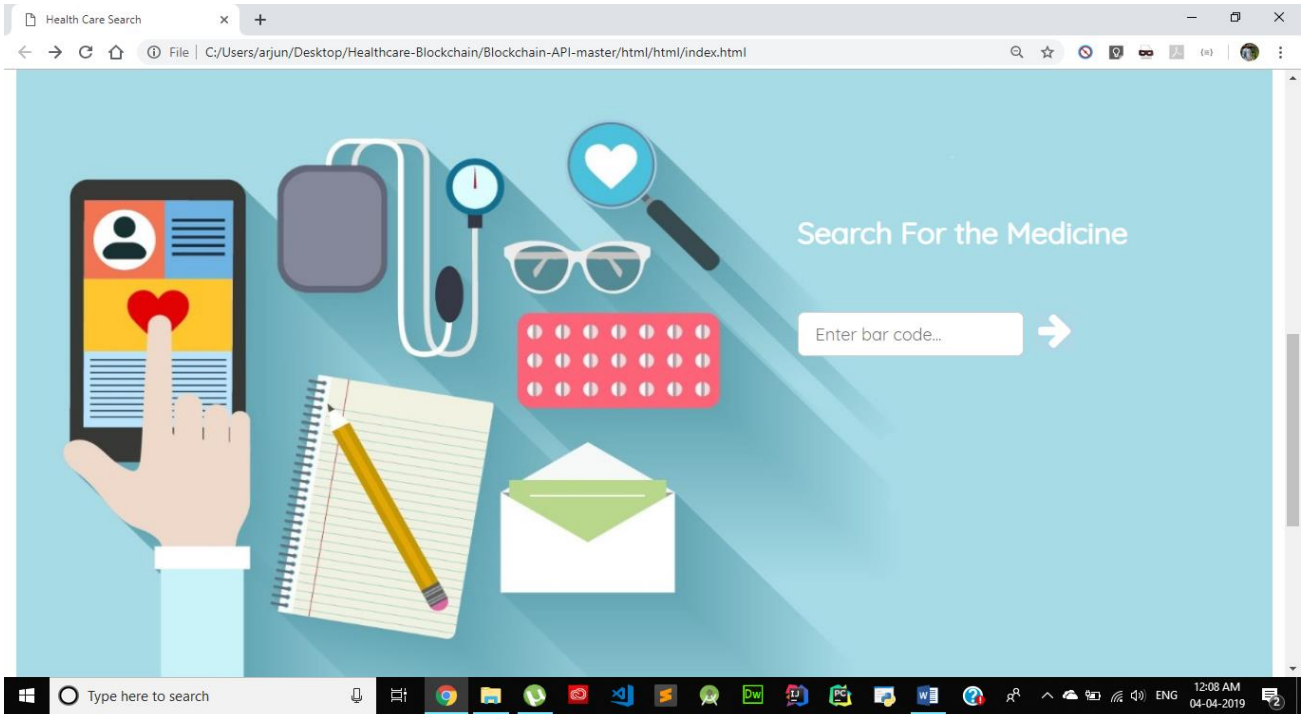
```
  });
});

module.exports = app;
```

MODULE SCREENSHOTS

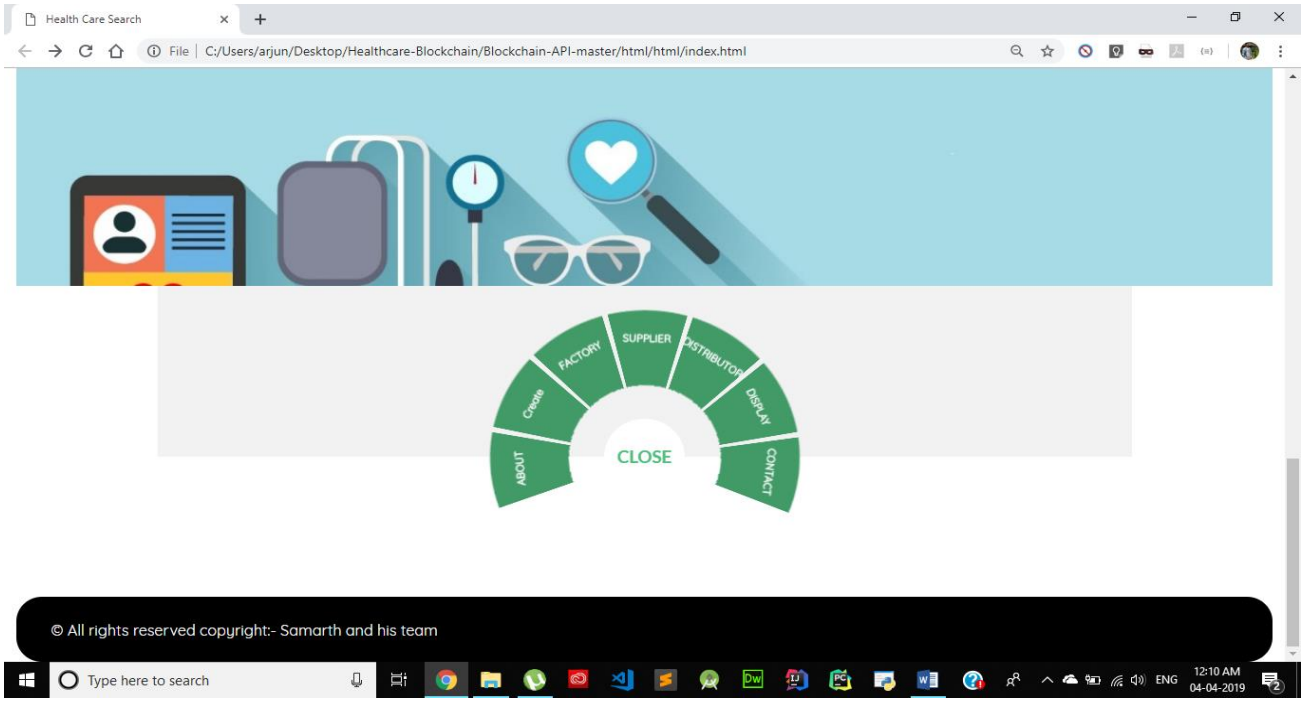
>Home Page



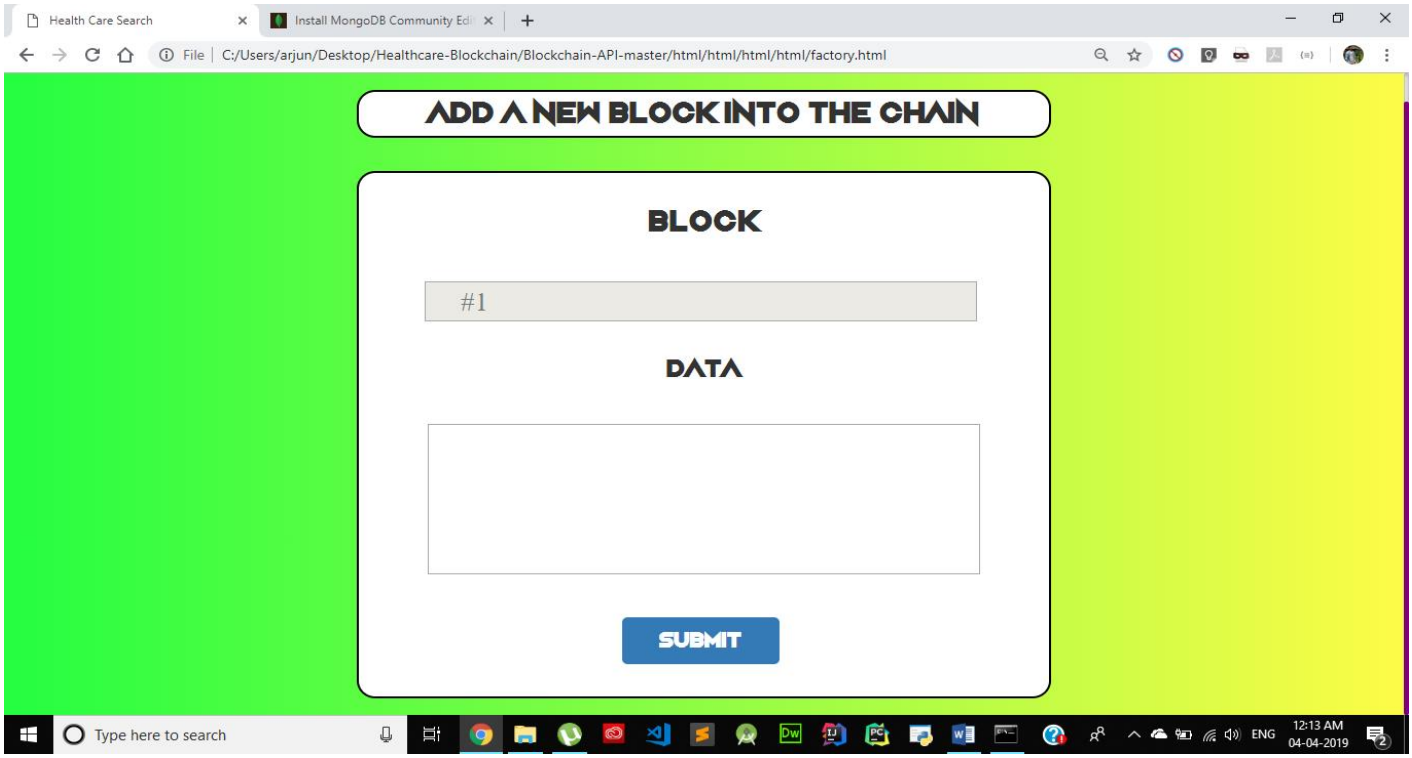
>Search Medicine



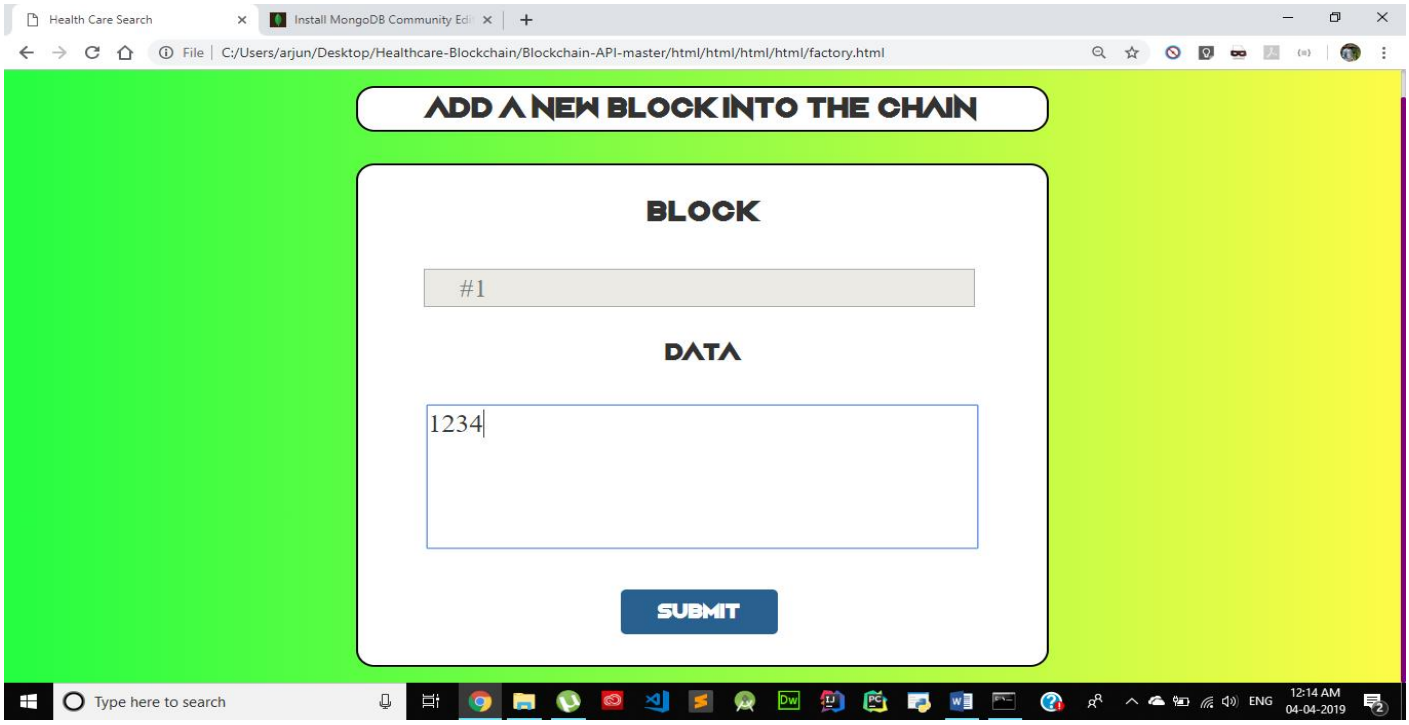
>Menu



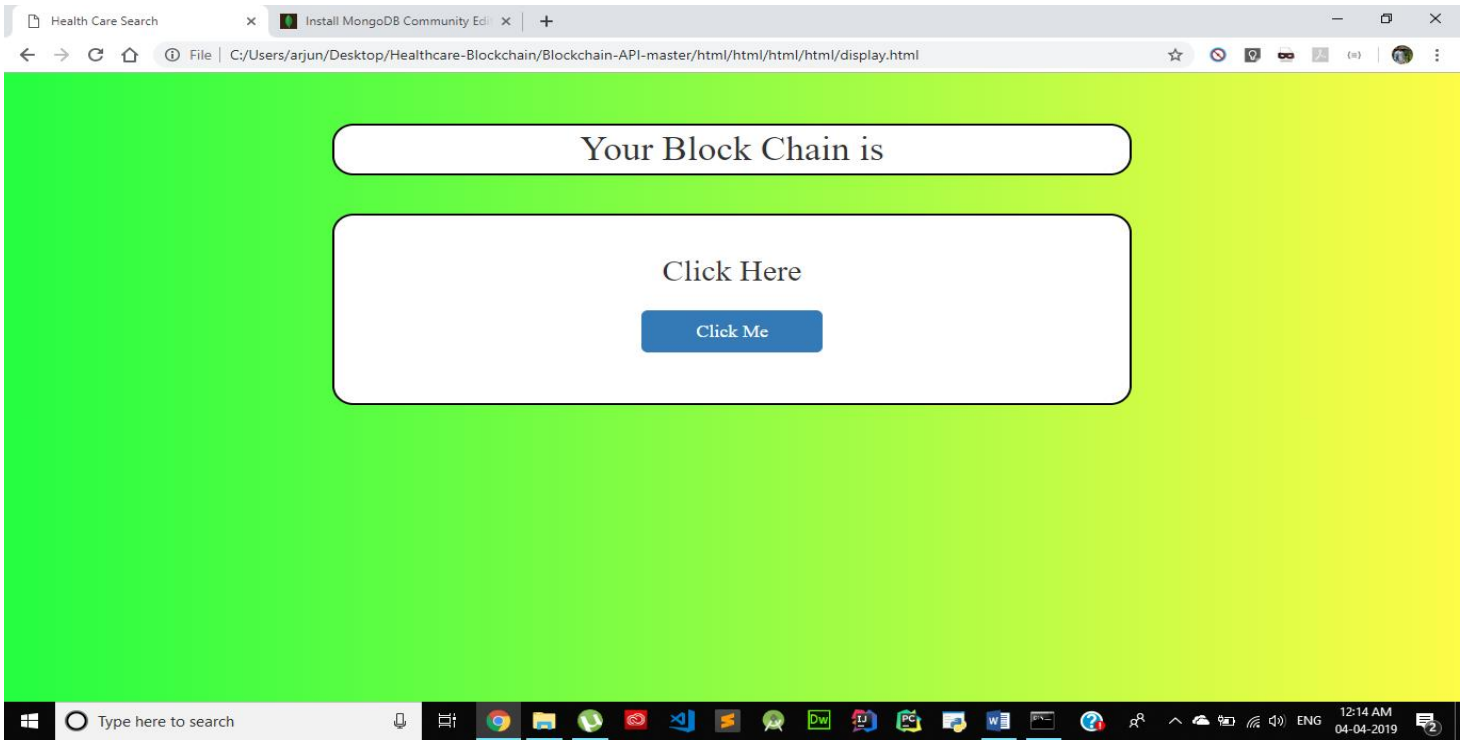
>Creation of a Block



>Entering of Data into the Block



>Display the whole BlockChain



TEST CASES

STEP	TEST STEPS	TEST CONDITION	EXPECTED RESULT	ACTUAL RESULT	STATUS
1	CREATING BLOCKCHAIN	CHECKING IF ATLEAST ONE BLOCK EXISTS	BLOCK EXISTS	BLOCK DOES EXIST	PASS
2	CREATION OF BLOCKCHAIN	HASHCODES OF BLOCKS ARE CONNECTED OR NOT?	HASHCODE OF ONE BLOCK CONNECTS TO THE HASH CODE OF NEXT BLOCK	HASHCODES ARE CONNECTED	PASS

3	BARCODE ENTERING	IS BARCODE CORRECT?	BARCODE IS NOT CORRUPT	BARCODE IS NOT CORRUPT	PASS
4	ENTER USERNAME	ENTER VALID USERNAME AND VALID EMAIL	LOGIN SUCCESS	LOGIN SUCCESS	PASS
5	ENTER VALID USERNAME AND INVALID PASSWORD	IS LOGIN SUCCESSFUL?	INCORRECT PASSWORD	INCORRECT PASSWORD	FAIL
6	VERIFY THE PREDICTION OF MEDICINE WHICH DOESN'T EXIST?	IS PREDICTION SUCCESSFUL?	MEDICINE DOEDN'T EXIST	MEDICINE DOESN'T EXIST	FAIL
7	VERIFY THE AVAILABILITY OF DOCTORS	ARE DOCTORS SHOWN?	USER IS ABLE TO ACCESS THE LIST	USER IS ABLE TO ACCESS THE LIST	PASS

RESULT ANALYSIS AND DISCUSSION

- The developed web application has the ability to provide the medicine just by entering the name. The customer gets to know the nearest medical stores from which he can order the medicines. This makes the app highly efficient for the customer.
- The medicine can be authenticated just by entering the barcode. The customer gets all the company and manufacturing details, making the authentication quite easy and efficient.
- The patients can get the details of the free slots of the doctors along with their registration fee and location through the app.
- The application can be easily handled even by novice users and its interface is quite user friendly.
- Block-chain technology is getting eminence in the market now a days and it is being used in various sectors. This technology can play a great role in decentralizing the health sector.

REFERENCES

1. Deep Shift. Technology Tipping Points and Societal Impact", *World Economic Forum*, September 2015, http://www3.weforum.org/docs/WEF_GAC15_Technological_Tipping_Points_report_2015.pdf

2. G. Prisco, *The Blockchain for Healthcare: Gem Launches Gem Health Network With Philips Blockchain Lab*, April 2016, https://bitcoinmagazine.com/articles/the-blockchain-for-heathcare-gem-launches-gem-health-network-with-philips-blockchain-lab-1461674938

3. P. B. Nichol, *Blockchain applications for healthcare*, March 2016, [online] Available: http://www.cio.com/article/3042603/innovation/blockchain-applications-for-healthcare.html

4. P. Taylor, *Applying blockchain technology to medicine traceability*, April 2016, [online] Available: https://www.securingindustry.com/pharmaceuticals/applying-blockchain-technology-to-medicine-traceability/s40/a2766/#.V5mxL_mLT

5. G. Dwyer, "The economics of Bitcoin and similar private digital currencies", *Journal of Financial Stability*, vol. 17, pp. 81-91, April 2015